IN THE CLAIMS

Please amend the claims as follows:

1. (Currently amended) A method ~~of identifying reusable computation units~~ comprising:

mapping n-dimensional architectural state vectors into a plurality of one-dimensional symbols;

arranging the plurality of one-dimensional symbols into phrases of text; and

identifying recurrent phrases of text as reusable computation units.

2. (Original) The method of claim 1 wherein mapping comprises:

traversing a software block in program execution order;

assigning new symbols as previously un-encountered architectural state vectors are encountered; and

assigning previously assigned symbols as previously encountered architectural state vectors are encountered.

3. (Original) The method of claim 2 wherein assigning new symbols comprises assigning consecutive integers such that each new symbol is assigned a value that is one greater than a previously assigned value.

4. (Original) The method of claim 1 wherein arranging comprises arranging symbols in program execution order.

5. (Original) The method of claim 4 wherein architectural state vectors include live-in states and live-out states for individual processor instructions.

6. (Original) The method of claim 1 wherein identifying comprises compressing the phrases of text to find a plurality of recurrent phrases.

7. (Original) The method of claim 6 wherein compressing comprises compressing the phrases of text using a lossless compression algorithm.

8. (Original) The method of claim 7 further comprising generating at least one trigger for a conjugate processor, the at least one trigger to implement complete reuse.

9. (Original) The method of claim 6 wherein compressing comprises compressing the phrases of text using a lossy algorithm.

10. (Original) The method of claim 9 further comprising generating at least one trigger for a conjugate processor, the at least one trigger to implement partial reuse.

11. (Original) The method of claim 6 wherein identifying further comprises correlating the plurality of recurrent phrases to identify reusable computation units.

12. (Original) The method of claim 1 further comprising annotating the reusable computation units in a program binary to cause a processor to memorize reuse instances.

13. (Currently amended) A computer-implemented method ~~of identifying reusable computation units~~ within an executable program comprising:

creating an execution trace of the executable program;

compressing the execution trace to find recurrent portions thereof; and

identifying the recurrent portions of the execution trace as reusable computation units.

14. (Original) The computer-implemented method of claim 13 wherein creating an execution trace comprises:

executing the executable program; and

mapping architectural states of the executable program into symbols.

15. (Original) The computer-implemented method of claim 14 wherein mapping architectural states into symbols comprises:

assigning integers to n-dimensional architectural state vectors such that each new n-dimensional architectural state vector is assigned an integer that is one greater than the last integer assigned.

16. (Original) The computer-implemented method of claim 15 wherein the n-dimensional architectural state vectors include information from processor instructions, live-in states, and live-out states.

17. (Original) The computer-implemented method of claim 13 wherein compressing comprises:

applying a compression algorithm that identifies an editing distance between similar recurrent phrases.

18. (Original) The computer-implemented method of claim 15 wherein identifying recurrent portions of the execution trace comprises:

correlating the dictionary of recurrent phrases with the executable program.

19. (Original) The computer-implemented method of claim 13 wherein compressing comprises:

applying a compression algorithm that identifies a dictionary of recurrent phrases.

20. (Original) The computer-implemented method of claim 19 wherein identifying recurrent portions of the execution trace comprises:

correlating the dictionary of recurrent phrases with the executable program.

21. (Original) The computer-implemented method of claim 20 further comprising annotating the reusable computation units in the executable program.

22. (Original) The computer-implemented method of claim 20 further comprising generating conjugate processor triggers to exploit reusable computation units in the executable program.

23. (Currently amended) An article having a computer-readable medium, the computer-readable medium having stored thereon instructions for a method ~~of identifying reusable computation units, the method~~ comprising:

compressing phrases of symbols that represent architectural states to identify recurrent phrases of symbols; and

correlating recurrent phrases of symbols with an executable program to identify reusable computation units within the executable program.

24. (Original) The article of claim 23 further comprising:

generating the phrases of symbols by mapping n-dimensional architectural states to one-dimensional symbols.

25. (Original) The article of claim 24 wherein mapping comprises:

executing an executable program; and

assigning an integer to each unique n-dimensional architectural state vector representing a processor instruction, live-in states, and live-out states.

26. (Original) The article of claim 25 wherein executing the executable program comprises generating a program trace that includes the n-dimensional architectural state vectors.

27. (Original) The article of claim 23 wherein compressing phrases of symbols comprises applying a lossless coding algorithm to the phrases of symbols.

28. (Original) The article of claim 27 wherein the method further comprises generating instruction triggers for a conjugate processor to implement complete reuse.

29. (Original) The article of claim 23 wherein compressing phrases of symbols comprises applying a compression algorithm that identifies an editing distance between similar phrases of symbols.

30. (Original) The article of claim 29 wherein the method further comprises generating instruction triggers for a conjugate processor to implement partial reuse.